Document Summarization on Long-Span Task with Content Selection and Attention Mechanism

Huiruo Zou and **Ying Yuan**
zouhr@umich.edu
yyyuan@umich.edu

Abstract

Transformers have been widely used in Documentation Summarization Tasks. However, it has limitations on long-span documents because of its full self-attention mechanism, which would require plenty of computing resources. One approach to deal with this problem is to change the attention layers to focus on less amount but more important information. Inspired by other works on attention, we modified on Bidirectional and Autoregressive Transformer (BART), one of the most successful models on text summarization, to adapt different attention mechanism with multi-task content selection, trying to improve the performance without too much computing resources.

Our result proves the feasibility and advancement of utilizing sparse global-local self-attention within BART model and using the explicit content selection for input data.

1 Problem Statement

Automatic text summarization is a language generation task in Natural Language Processing (NLP) that aims to create a summary that represents the most important information and salient ideas from the original source text while preserving the overall meaning. This task has been studied for about 70 years. There are two types of text summarization task: abstractive text summarization and extractive summarization (Pilault et al., 2020). Abstractive summarization is an approach to generate a concise summary by paraphrasing using new sentences, while extractive summarization extracts sentences directly from the original document and stacks them to generate summary. Abstractive summarization task, especially the abstractive summarization task on long documents, is more advanced but has higher hardware demand.

In the past two years, with the Transformer, a deep learning model, being proposed, a number of Transformer-based methods have been developed for abstractive summarization and have made great progress (Zaheer et al., 2021). The community has achieved some desired results with pre-trained Transformers on summarization. However, at the same time, Transformer-based models are not good at processing long input length. Their full attention mechanism causes quadratic dependency growing with the longer input sentence (Beltagy et al., 2020). High memory and compute requirement caused by long-span dependencies have been an issue for solving. Therefore, some large pre-trained models may not be adaptable for long document data to have satisfied performance.

Problem in memory and compute requirements caused by long-span summarization is an interesting and complicated area for researchers to explore. Thus, in general speaking, we aim to propose a model to mitigate this problem while also maintain or even improve the generation quality. Increasing data-efficiency is a good main direction to solve this problem. From this perspective, we come up with several possible solutions. Instead of exhausting resources to figure out all potential dependencies, modifying the full attention that causes quadratic dependency so that the new attention could reduce quadratic dependency to linear could be a possible solving method. Also, focusing on selecting context with high importance could be another possible solving method. Recently, a system, LoBART, using local self-attention instead of full attention in BART model motivates our project idea (Manakul and Gales, 2021). It also modifies the implicit content selection to explicit content selection to reduce memory and compute requirements. To be specific on our particular problem, we aim to investigate more efficient attention pattern within BART model, a denoising autoencoder for pretraining sequence-to-sequence models (Lewis et al., 2020), while using explicit content selection for the text data preprocessing.

In this project[1], we tackled the long-span dependencies in abstractive summarization with methods: local self-attention; sparse self-attention; sparse global-local self-attention; beam search; and explicit content selection. We combined techniques mentioned in our new architecture, trained the models, generated new summarizations, and evaluated our results. We reproduced the LoBART model as the baseline model and compared the results with our newly proposed model.

Challenges exist in several aspects in this project, we faced challenges in correctly reproducing LoBART, which was implemented based on the Huggingface NLP package. Due to this frequently updated package, changing on the package source code is usual. Also, because of the complexity of the transformer model, it is difficult to modify self-attention within BART. Challenge also exists in high training cost. The training had high demand in GPU and memory due the complexity of the model and the large dataset increase the workload in training and hyperparameter tuning. With the configuration provided by author, using RTX 3090 GPU, the training of LoBART baseline requires more than 30 hours; and the generation using the trained model requires about 6 hours.

Finally, the result proves the feasibility of utilizing different attention patterns within BART model and using the explicit content selection for input data. Quantitative evaluations reflect improvement with using sparse self-attention and sparse global-local self-attention.

2 Related Work

Transformer-based models, such as BART, have been one of the most successful models for text summarization. Transformer is a deep learning model that has separate encoder and decoder architecture and relies on self-attention mechanism to weight the significance of the data. However, transformer-based models are not good at processing growing input length due to their full self-attention mechanism. The long-span dependencies cause challenge in memory and compute requirements. These years, to tackle this problem aroused by long-span dependencies, several systems modify the pattern of their self-attention to scale the linearly instead of scaling quadratically.

[1]Code is available at https://github.com/YinnnYwY/LongAttn.git

Zaheer et al. (2021) introduce a block sparse attention mechanism in their system that could reduce the quadratic dependency to linear. Their block sparse attention with block size of 2 is a combination of the random attention with 1 random token, sliding window attention (namely, local attention) with 3 local neighboring tokens, and global attention with 1 global token attending on all parts of the sequence. Bigbird attention reduces the complexity with local attention. At the same time, it maintains full sequence representations using the global attention. They use block sparse attention on Bidirectional Encoder Representations from Transformers (BERT) to handle long length input. It inspired us that we could adapt its attention mechanism on our BART-based model to see if it boosts the performance.

Beltagy et al. (2020) also design and implement another sparse global-local self-attention mechanism: it combined sliding window attention (namely, local attention) for local context with task motivated global attention at a certain tokens to improve the performance on long-document. Longformer attention reduces the complexity to linear with local attention. Meanwhile, it maintains full sequence representations using the global attention. They use this sparse attention on BERT style models to handle long length input.

Other systems include the modification on content selection to highlight the importance.

Cao and Wang (2021) has shown that adding a masking to attention head on encoder-decoder attentions in the decoder of BART-based model would effectively inform content selection and help generate better abstract summary. Additionally, the masking is only used at inference time without model modification. They really focus on how to extract importance from source by the BART-based large pre-trained model on regular text data (like CNN/DM (Hermann et al., 2015)). They adapt attention head masking to improve content selection to highlight the essential part of the text for learning. Although this method aims to solve the text summarization from text data with regular length, the idea could be adopted. However, we are not able to combine attention head masking. Please see the discussion session for the reason.

Manakul and Gales (2021) also incorporates content selection in the architecture. They find out that adding local self-attention to the encoder part of the BART-based model and adopting explicit content

selection could outperform baseline BART model on long-span summarization task. They propose a Multitask Content Selection (MCS) method as the testing-time content selection and use oracle methods as the training-time content selection. We could concentrate on salient text to reduce complexity.

Accordingly, we would like to experiment on improving long-span data summarization with Multitask Content Selection (MCS) method and attention head masking, and performance improvement with local attention and other attention mechanism.

3 Approaches

Transformer-based models, such as BART, are trending in NLP for article summarization. Transformer is a deep learning model that has separate encoder and decoder architecture and relies on self-attention mechanism to compute representations of its inputs and outputs. One of transformers, the Bidirectional and Autoregressive Transformer (BART), is a state-of-the-art model to combine Bidirectional Encoder with an Autoregressive decoder into one Sequence-to-Sequence model (Lewis et al., 2020). It has been widely used for abstractive summarization in NLP. The architecture of the transformer is shown in Figure 1.

In our study, we adopt BART as the baseline. Due to the attribute of long dependencies for long-span text, the original self-attention in BART could require large amount of computing resources, which is not very efficient and time consuming. In our project, we implemented a BART-based architecture for long-document summarization which incorporates explicit multitask content selection (MCS) (Manakul and Gales, 2021), trying to diminish calculations among tokens. The BART model part is improved by utilizing two methods: local self-attention (Manakul and Gales, 2021) and multiple other attention mechanism.

3.1 Content Selection

3.1.1 Multitask Content Selection (MCS)

The explicit MCS model we adopt has shown improvement over BART baseline with implicit content selection model on long-document summarization. The result produced by Manakul and Gales (2021) indicates that explicit MCS model is able to reduce memory and compute requirements.

This model extends the sentence-level GRUs in hierarchical encoder-decoder architecture with

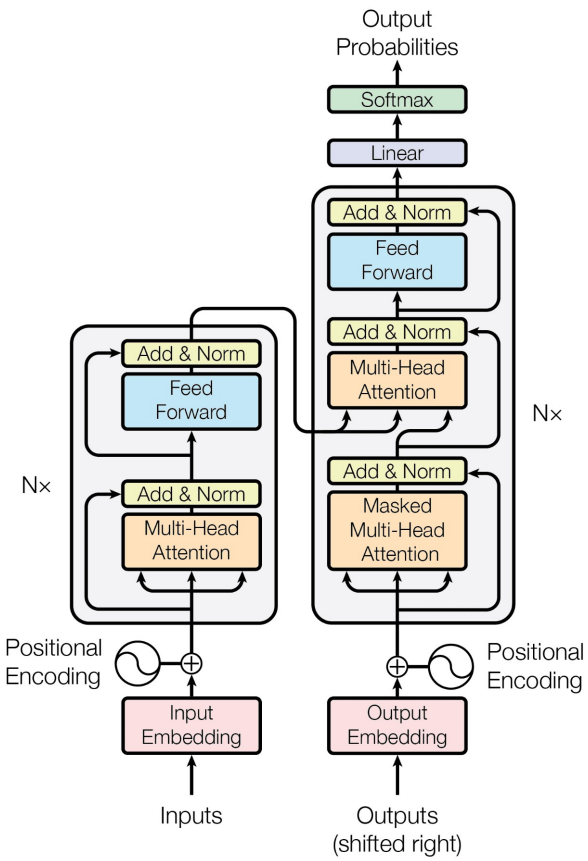

Figure 1: Transformer Architecture in Detail (Vaswani et al., 2017)

two tasks: sentence-level encoder-decoder attention mechanism; and extractive labelling modules and creating labels heuristically, to improve performance and reduce memory.

MCS is used as the testing time (namely, inference time) content selection. Before decoding, the data are processed by MCS at first to select out important content. Then the processed result will be feed into our model to decode.

3.1.2 Oracle

Oracle is a ground-truth based method for training-time content selection. With the availability of ground-truth summary and the similarity measure, the importance of the sentence could be re-ranked, truncated, and sorted. Manakul and Gales (2021) propose several variant of oracle methods to extend the input selected by no-padding oracle method. Results show that oracle method with pad by random unselected sentences and keeping the original sentence order outperforms because it brings more diversity to the abstractive model.

3.2 BART

We use fine-tuned BART model (Lewis et al., 2020) and apply modification on it. To reduce the complexity, the self-attention mechanism in the encoder of BART is modified to local multi-head self-attention and other attention mechanism. To highlight salient content and block attentions to unimportant tokens from the source.

3.2.1 Local Attention

Local attention (namely, sliding window attention) compared to full attention, only focus on the information provided by neighbors to reduce computational complexity. It applies a fixed-size window attention around each token. With the fixed-size W and the input sequence length n, the complexity of local attention is $O(n \times W)$ instead of $O(n^2)$. However, it is not sufficient to capture all the context with local attention. Manakul and Gales (2021) implemented the BART model with local attention. We will use it as the baseline to compare with other BART variants we implemented. Fig 2 shows the illustration of full attention and the local self attention with window size (W) of 9.

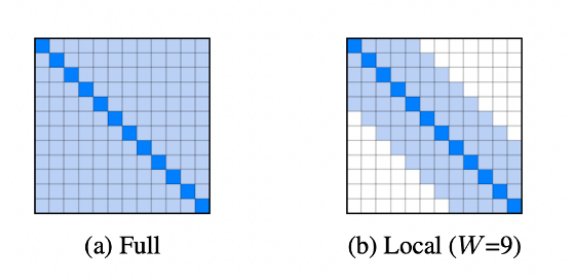

Figure 2: Full and Local Attention
(Manakul and Gales, 2021)

3.2.2 Random Attention

Besides local self attention, we also try to implement it with local random self-attention. For each token, it only focus randomly on others. Figure 3 shows a random local attention created by Big-Bird (Zaheer et al., 2021) with 1 random token and block size of 2. Random attention enables small average path length between nodes. However, it is not sufficient to capture all the context and the performance is worse than BERT baseline. Thus, we will not implement the BART model with only random attention, and we will investigate the attention that combines random attention and other patterns.

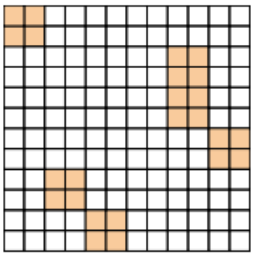

Figure 3: Random Attention
(Zaheer et al., 2021)

3.2.3 Sparse Attention

Another attention mechanism that we implement is Sparse Attention. Compared with Local Attention, which accounts for the full W neighbors, our Sparse Attention only consider total $\frac{W}{2}$ neighbors, taking every other nodes of local attention with window length of W.

3.2.4 Global Attention

A set of tokens with global attention attends to all parts of the sequence and all the tokens in the sequences attend to each token in the set. Full sequence representations would be maintained by using the global attention. Global attention is always combined with other attention patterns. We will investigate the attention that combines global attention and other patterns. Figure 4 shows a global attention created by Big-Bird (Zaheer et al., 2021) with 1 global token and block size of 2.

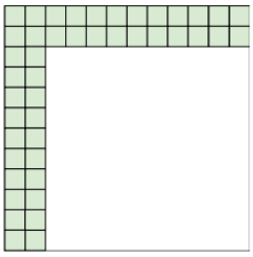

Figure 4: Global Attention
(Zaheer et al., 2021)

3.2.5 Bigbird Attention (Global-local)

Bigbird attention with block size of 2 is a combination of the random attention with 1 random token, sliding window attention (namely, local attention) with 3 local neighboring tokens, and global attention with 1 global token attending on all parts of the sequence. It reduces the complexity to linear with local attention. At the same time, it maintains full sequence representations using the global at-

tention. Figure 5 shows a Bigbird attention created by Big-Bird (Zaheer et al., 2021).

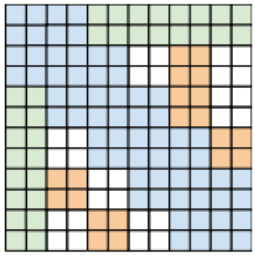

Figure 5: Bigbird Attention (Zaheer et al., 2021)

3.2.6 Longformer Attention (Global-local)

Longformer Attention is a sparse attention mechanism that combines sliding window attention (namely, local attention) for local context with task motivated global attention at a certain tokens to improve the performance on long-document. Longformer attention reduces the complexity to linear with local attention. Meanwhile, it maintains full sequence representations using the global attention. Figure 6 shows a Longformer attention created by Beltagy et al. (2020).

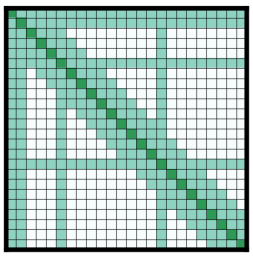

Figure 6: Longformer Attention (Beltagy et al., 2020)

3.3 Beam Search

Beam search is commonly used in NLP models. For decoder, instead of only finding the best token at each position as Greedy Search, Beam Search will find best "N" best sequences. Even though some times Beam Search may require more resources, but it could produce better results because it could consider in different sequence instead of single position.

We would like to see the performance affected by different beam size. So, we tried to decode the model with different beam length. We would like to see the number of chosen beam could affect the model performance.

3.4 Data Set

To compare our model with previous state-of-art BART-based model for long-document summarization, we utilize summarization datasets that are commonly used for this type of model. Originally we plan to use both Long-Span text dataset and regular dataset, for example, CNN/DM(Hermann et al., 2015)). However, due to limited computation resources and time constraints, we are not able to process and train over multiple giant datasets. We decided to mainly focus on only one scientific article dataset.

PubMed. The long and structured documents dataset of scientific articles contains article content, abstracts, and section names of over 100 thousand papers from PubMed repository. This dataset has been processed into training, validation, and test set. For PubMed dataset, the number of samples in each set is 119,924, 6,633, and 6,658 (Cohan et al., 2018). We use the abstract of each article as the ground-truth for evaluation.

3.4.1 Data pre-processing

We first use the method suggested by Manakul and Gales (2021) to extract article id, abstract, and full text from the raw dataset. Then, we utilize the oracle method with padding by random unselected sentences as training-time content selection. Next, we feed the data into pre-trained MCS model to select out salient information and save it as the input to our model for evaluation.

3.5 Experiment

3.5.1 Beam Search

We tried several different beam search size on the paper-proposed model. We compared ROUGE scores between our summarizations and those of the paper. During the training, however, we could not reproduce the model created by Manakul and Gales (2021) directly, because it trains the model in 160,000 steps, which requires to run on our GPU for over 30 hours. Accordingly, training 160,000 steps seems unreasonable for us to reproduce. Then, we reduced the steps to 5,000 and started to train the model with {4, 6, 8} beam size.

3.5.2 Approach 1:

Oracle + Local Attention + MCS + Beam search
For the training time content selection, we use oracle method with padding by random unselected sentences. For the testing time content selection, we use MCS method.

To reproduce the BART model with local attention, we set the window size of local attention to 1024 and set the maximum input length to 4096. We decoded with different beam size, {4, 6, 8}, to find the best beam size for decoding. Due to limited computing resources and time, we are only able to use train 5,000 steps compared to the 160,000 steps trained by previous paper Manakul and Gales (2021). We only test on 1,000 samples, while previous paper test on 6,658 examples.

After the best beam size is tested, we train the BART model with local attention for 20,000 steps and test it on {1,000, 2,000, 3,000, 4,000, 5,000, 6,000, 6,658} examples with beam size of 4.

3.5.3 Approach 2:

Oracle + Sparse Attention + MCS

For the training time content selection, we use oracle method with padding by random unselected sentences. For the testing time content selection, we use MCS method.

To reproduce the BART model with Sparse attention, we set the window size of local attention to 1024 and set the maximum input length to 4096. Due to limited computing resources and time, we are only able to use train 20,000 steps compared to the 160,000 steps trained by previous paper Manakul and Gales (2021). We tested it on {1,000, 2,000, 3,000, 4,000, 5,000, 6,000, 6,658} examples with beam size of 4.

3.5.4 Approach 3:

Oracle + Longformer Attention + MCS

For the training time content selection, we use oracle method with padding by random unselected sentences. For the testing time content selection, we use MCS method.

To reproduce the BART model with Longformer attention, we set the window size of local attention to 1024 and set the maximum input length to 4096. Due to limited computing resources and time, we are only able to use train 20,000 steps compared to the 160,000 steps trained by previous paper Manakul and Gales (2021). We tested it on {1,000, 2,000, 3,000, 4,000, 5,000, 6,000, 6,658} examples with beam size of 4.

4 Evaluation

We use the average of automatic ROUGE-1,2,L recall scores (Lin, 2004) to test the model. ROUGE-1 reflects the unigrams overlap between the ground-truth summary and generated summary. ROUGE-2 reflects the bigrams overlap between the ground-truth summary and generated summary. ROUGE-L measures overlap based on longest common subsequence (LCS). The formula of ROUGE recall is:

$$\frac{number\ of\ overlapping\ words}{total\ words\ in\ reference\ summary} \quad (1)$$

Due to the limitation on time and budget, we will not implement human evaluation. During training, we use Cross-Entropy as the loss function.

4.1 Beam Search

At first, we tried several different beam search sizes on the paper-proposed model. We compared ROUGE-1,2,L recall scores between LoBART models with different beam size, and the results are shown in Table 1. As a result, we choose beam size 4 for future decoding.

#Beam	ROUGE-1	ROUGE-2	ROUGE-L
4	**36.407**	**11.907**	**25.967**
6	35.955	11.476	25.567
8	36.205	11.627	25.766

Table 1: Performance on Beam Search

4.2 Approaches

4.2.1 Local Attention (Approach 1)

The ROUGE-1,2,L recall scores of the BART model with local attention by the size of testing data set are shown in Table 2.

#Test	ROUGE-1	ROUGE-2	ROUGE-L
1000	39.438	15.344	28.289
2000	39.908	15.551	28.650
3000	39.666	15.340	28.423
4000	39.417	15.063	28.176
5000	39.378	15.063	28.163
6000	39.459	15.138	28.190
6658	39.477	15.145	28.226

Table 2: ROUGE table on BART with Local attention

We trained LoBART with different training steps, 5000 and 20000, and decoded with beam size of 4. Manakul and Gales (2021) trained LoBART with 160,000 step and decoded with beam size of 4. The ROUGE-1,2,L recall scores of the BART model with local attention by the training steps are shown in Table 3.

#Training	ROUGE-1	ROUGE-2	ROUGE-L
5000	36.407	11.907	25.967
20000	39.477	15.145	28.226
160000	**48.06**	**20.96**	**43.56**

Table 3: ROUGE table on LoBART

The cross entropy based training loss by training steps is plotted as Figure 7.

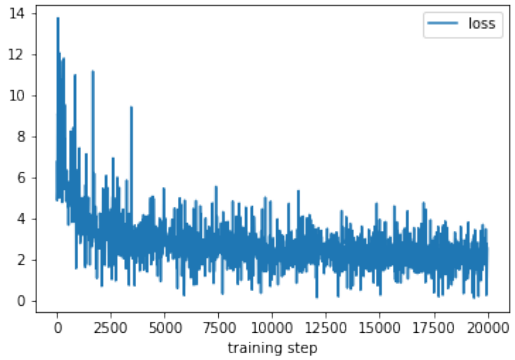

Figure 7: Local Attention

4.2.2 Sparse Attention (Approach 2)

The ROUGE recall scores of the BART model with sparse attention by the size of testing data set are shown in Table 4.

#Test	ROUGE-1	ROUGE-2	ROUGE-L
1000	39.014	15.140	28.163
2000	39.377	15.162	28.290
3000	39.133	14.883	28.070
4000	38.986	14.629	27.909
5000	38.955	14.633	27.912
6000	39.046	14.705	27.988
6658	39.052	14.714	28.001

Table 4: ROUGE table on BART with Sparse attention

Figure 8 shows the training error log.

4.2.3 Longformer attention (Approach 3)

The ROUGE recall scores of the BART model with sparse attention by the size of testing data set are shown in Table 5.

Figure 9 shows the training error log.

4.2.4 Approaches Comparison

The ROUGE recall scores of the BART model with different attention trained with 20,000 steps, decoded with beam size of 4, and tested with full test dataset, are shown in table 6.

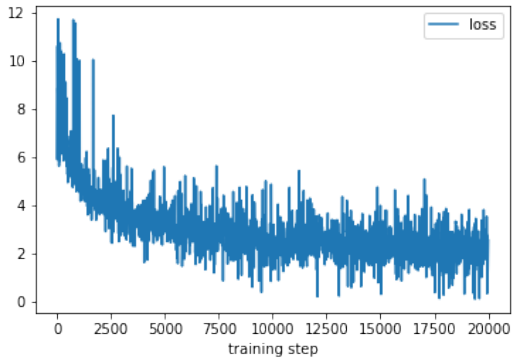

Figure 8: Training Log for Sparse Attention

#Test	ROUGE-1	ROUGE-2	ROUGE-L
1000	40.206	15.967	28.931
2000	40.431	15.967	29.097
3000	40.206	15.967	28.920
4000	39.942	15.967	28.688
5000	39.893	15.967	28.704
6000	39.985	15.967	28.765
6658	39.958	15.967	28.766

Table 5: ROUGE table on BART with Longformer attention

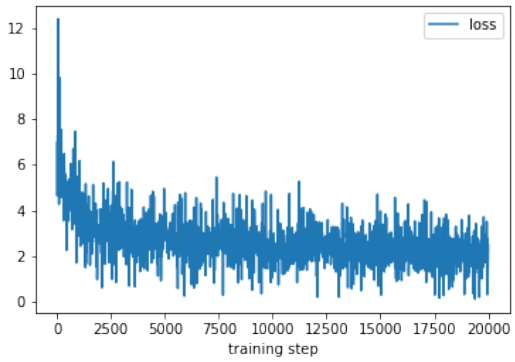

Figure 9: Training Log for Longformer

Attention	ROUGE-1	ROUGE-2	ROUGE-L
Local	39.477	15.145	28.226
Sparse	39.052	14.714	28.001
Longformer	**39.958**	**15.515**	**28.766**

Table 6: ROUGE table on BART with different attention

Figure 10 shows ROUGE-1 score of BART model with different attention patterns by test size.

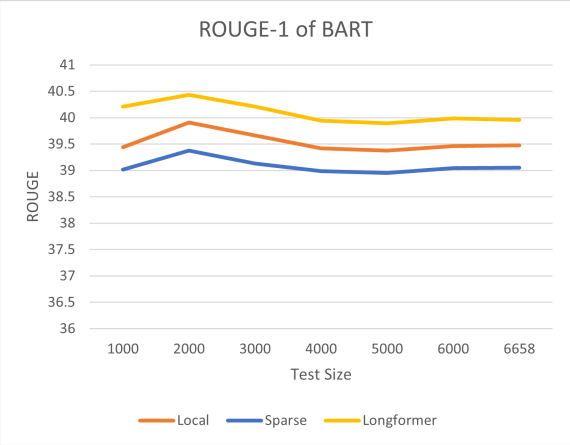

Figure 10: ROUGE-1 score of BART variants

Figure 11 shows ROUGE-2 score of BART model with different attention patterns by test size.

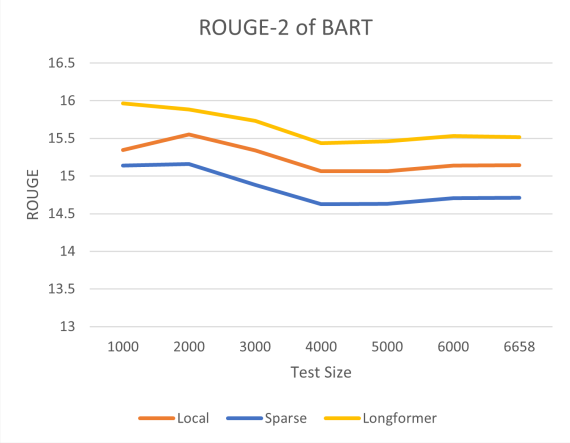

Figure 11: ROUGE-2 score of BART variants

From the results, we could tell that our Longformer model out-perform the baseline. However, the model with sparse attention has a worse summarization quality. We will discuss about affect of the number of training step in the next Section.

5 Discussion

5.1 Attention Head Masking

We first were really interested in attention head masking with content selection since we assumed that selecting important tags with MCS could really improve the performance on summarization quality and computing time (Cao and Wang, 2021). However, it turns out that this approach is not every easy to adapt on our model. Because attention head masking with content selection requires a tagger to pin out salient information and pass into the model, which is a new feature that we need to add from scratch to our existing model.

Initially we want to reproduce their work (Cao and Wang, 2021). We tried to set up the environment on both local GPU and GreatLakes. However, none of them works. The training could cause *CUDA out of memory* issue and There are some wired bugs happening in the source code of the packages.

After thoroughly studied (Cao and Wang, 2021) source code, we find out that in order to combine the masking approach with our model, it requires plenty of code editions in the source code of models for multiple structures. Studying the source code already took us weeks while we were trying to implement our model. We do not have time to dig deeply into every function and make sure it works properly for the tagger and masking. Thus, we switched our direction into discovering the effect on different attention mechanism on our model.

5.2 Training Step

One of the biggest challenge for our project is training time. At beginning, we did not realize that processing on long-span data set requires more time than regular data set for either encoding or decoding phase. As for reproducing, due to limited time, we could not train the model with the training steps exactly as same as the paper did. So, as the result shown in the Table 3 in Evaluation section are not super close to the best result that the paper suggested. It is clear that the model trained with more training steps would the achieve better performance. The LoBART trained with 20,000 steps has **9%** higher ROUGE score than the LoBART trained with 5,000 steps. However, 20,000 training step is relatively small and far from overfitting. We are not sure about what training step would lead to overfitting.

Also, during our other training, this problem is still there. While we reduce the training step, the performance affect in some way. However, according to Figure 7, 8 & 9, the loss for each training step shows that when we do train with more steps, the loss does not decrease significantly. It could be one interesting fact to discuss about the training steps' effect on the model performance.

5.3 Testing Data Size

Due to resource limitation, we did not use on the whole testing data set at first. It is another interesting fact that when we adjust the number of data we choose to test our model performance, the ROUGE score changes. Shown in Figure 10 & 11, more testing data actually being evaluated could decrease the ROUGE score. So it is possible that in our first model whose testing data is only 1000 compared to the whole test set, 6,658, The evaluation ROUGE score does not fully represent the summarization quality. With more testing data, it may require the model to have a bigger scope, which may affect the scores on some unfamiliar data.

5.4 Attention Mechanism

After trying to reproduce the model suggested by the paper with local self-attention of window size 1024, we would like to implement other attention mechanism. The ROUGE-1 score of the baseline tested with full test dataset is **39.477**.

5.4.1 Sparse Attention

The first attention we chose is sparse attention. In the beginning, we only want to adapt other's sparse attention layer to our baseline model. However, we realize that different architecture uses different model interface, which may cause some parameters mis-alignment. Instead of wasting time on changing interfaces, we approach to change the attention source code directly. So instead of using the whole local attention size, we implemented a attention layer with one mask in between each weight.

After training it for 20,000 steps, we got ROUGE-1 score for the performance is **39.052**. The new model with our sparse attention does not have a better performance than the baseline. Shown in Figure 10 & 11, sparse attention obtains lowest ROUGE-1,2 score among three approaches. In the sparse attention, even though we still use the window size of 1024, but we mask some information out. It is possible that some information with higher importance are filtered out so the decoder could extract weights with lower priority, affecting the summarization quality.

5.4.2 Longformer Attention (Global-local)

To adapt the Longformer attention to the BART model, we choose to modify the source code of baseline model. The baseline only contains local attention, while the Longformer attention consists of not only local attention and but also task motivated global attention at some tokens. The realization of global attention is completely different from local attention. This increases the complication in implementing global attention and the combination. Also, we realize that different architecture uses different model interface, which may cause some parameters mis-alignment.

The trained BART model with Longformer attention outperforms the local attention and sparse attention. The ROUGE-1 score is **39.958**. We could see the comparison in ROUGE score from Figure 10 & 11. Local attention and sparse attention reduce the complexity to linear but they are insufficient in capturing all the context at the same time. Global attention mitigates this drawback because it maintains full sequence representations. As a result, Longformer attention balance the model complexity and the representation capacity. This is why Longformer attention gains better performance on summarization than local attention and sparse attention.

6 Conclusion

In conclusion, we adapted the original BART model with sparse attention and Longformer attention in this project. Explicit content selection was applied to training time content selection to reduce memory and compute requirements. Comparison with the LoBART, Longformer attention reduces the complexity to linear and maintains full sequence representation at the same time without losing any summarizartion quality; and sparse attention avoids quadratic dependency but is not able to capture all context that led to lower generation quality. However, due to the time and hardware limitation, we are not able to do hyperparameter tuning and train the models with same number of training steps as paper. Some potential future work may be hyperparameter tuning; training with more training steps; implementing attention head masking; and adopting various attention pattern.

7 Division of Work

7.1 Final Report

7.1.1 Sections for Ying Yuan

1. Abstract
2. Problem Statement
3. Related Work
4. Approaches
   (a) Content Selection

(b) BART
(c) Experiment
5. Evaluation Tables
6. Discussion
7. Conclusion

7.1.2 Sections for Huiruo Zou

1. Abstract
2. Problem Statement
3. Related Work
4. Approaches
   (a) Beam Search
   (b) Sparse Attention
   (c) Data Set
5. Evaluation Figures
6. Discussion

7.2 Implementation

7.2.1 Tasks for both team member

1. Set up environment for both Manakul and Gales (2021) and Cao and Wang (2021);
2. Reproduce results from the papers mentioned above as baseline;
3. Study the code base.

7.2.2 Tasks for Ying Yuan

1. Pre-process the pudMed data with Oracle for training;
2. Pre-process the pudMed data with MCS for decoding;
3. Adapt the Longformer attention into BART model;
4. Train the baseline model and BART with Longformer attention;
5. Decode trained models;
6. Compute the ROUGE score for trained models.

7.2.3 Tasks for Huiruo Zou

1. Try to adapt Multi-Head Masking into the current model
2. Adapt the sparse attention into BART model;
3. Train the BART with sparse attention;
4. Decode the summarization for each article with each model.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Shuyang Cao and Lu Wang. 2021. Attention head masking for inference time content selection in abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5008–5016, Online. Association for Computational Linguistics.

Ann Clifton, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, Gareth Jones, Jussi Karlgren, Ben Carterette, and Rosie Jones. 2020. 100,000 podcasts: A spoken English document corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5903–5917, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Linguistic Data Consortium and New York Times Company. 2008. *The New York Times Annotated Corpus*. LDC corpora. Linguistic Data Consortium.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All NLP tasks are generation tasks: A general pretraining framework. *CoRR*, abs/2103.10360.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Rosie Jones, Ben Carterette, Ann Clifton, Jussi Karlgren, Aasish Pappu, Sravana Reddy, Yongze Yu, Maria Eskevich, and Gareth J. F. Jones. 2020. Trec 2020 podcasts track overview. In *TREC*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Potsawee Manakul and Mark Gales. 2021. Long-span summarization via local attention and content selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*

and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 6026–6041, Online. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Christopher Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences.